Spacetime agnostic general relativistic ray-tracing through automatic differentiation

F. J. E. Baker,[1★] and A. J. Young[1]
[1]H. H. Wills Physics Laboratory, Tyndall Avenue, Bristol BS8 1TL, UK

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT
We introduce Gradus.jl, an open-source...

Key words: keyword1 – keyword2 – keyword3

1 INTRODUCTION

In the era of quantitative, precision observational tests of General Relativity in the strong field regime it is necessary to have a fast and flexible method to compute the observational properties of accreting black hole systems. We have developed an open-source integrator Gradus.jl[1] for this purpose. In the remainder of the paper we describe how the software works, comparing with previous work in the literature, and outlining the new capabilities of Gradus.jl.

Transfer functions (Cunningham 1975)

Julia is a high-performance... with SciML and DifferentialEquations.jl, a state-of-the-art ecosystem and workhorse for solving differential equations.

The trajectory of light in curved space may be determined by reformulating the Hamilton-Jacobi equations of motion as a first-order ordinary differential equation (ODE) system.

A second-order ODE system may alternatively be formulated directly from the geodesic equation; a method which is pedagogically simpler, but computationally more expensive than the first-order system, as either the full metric connection or derivatives of the metric must be explicitly implemented, else approximated at cost during runtime. With advancements in automatic differentiation, derivatives are cheap to compute, and consequently the second-order approach is tractable and both a parsimonious and spacetime agnostic method for computing geodesics.

2 NUMERICAL METHODS

Automatic differentiation (AD) is a computational method for calculating derivatives through the chain rule, relying on the chronological sequence in which mathematical expressions are calculated on computers to propagate derivatives through a function call. Forward mode AD is a specific method of accumulation that uses the algebra of dual numbers to track tangent components, first computing the result and dual for each sub expression in order to find the total derivative.

For simplicity, this section will focus on static, axisymmetric spacetimes in the Boyer-Lindquist coordinates, in which a metric may be expressed

$$g_{\mu\nu} = g_{tt}\mathrm{d}t^2 + g_{rr}\mathrm{d}r^2 + g_{\theta\theta}\mathrm{d}\theta^2 + g_{\phi\phi}\mathrm{d}\phi^2 + 2g_{t\phi}\mathrm{d}t\mathrm{d}\phi, \tag{1}$$

reserving greek indices to denote the four spacetime components, and latin indices for the three spatial components. We write partial derivatives with respect to the coordinates x^μ as $\partial_\mu := \partial/\partial x^\mu$, and distinguish Euler's number e and basis vectors e with italics.

2.1 Geodesic integration

The geodesic equation for coordinates x^μ is

$$\frac{\mathrm{d}^2 x^\mu}{\mathrm{d}\lambda^2} + \Gamma^{\mu}{}_{\nu\sigma}\frac{\mathrm{d}x^\nu}{\mathrm{d}\lambda}\frac{\mathrm{d}x^\sigma}{\mathrm{d}\lambda} = a^\mu, \tag{2}$$

where λ is the affine parameter parameterising the curve, and $a^\mu = 0$ is an external non-gravitational acceleration vector. The Christoffel symbols are

$$\Gamma^{\mu}{}_{\nu\sigma} = \frac{1}{2}g^{\mu\rho}\left(\partial_\nu g_{\rho\sigma} + \partial_\sigma g_{\rho\nu} - \partial_\rho g_{\sigma\nu}\right), \tag{3}$$

defined solely by components of the metric and derivatives thereof. Note that the Christoffel symbols need only the metric and a sparse Jacobian of the metric to be determined. With forward mode AD, these quantities may be calculated simultaneously, both accurately and efficiently, and, along with (1), one can exploit $\partial_t g_{\mu\nu} = \partial_\phi g_{\mu\nu} = 0$ to reduce the number of operations for different spacetime classes.

A second-order system requires some initial x^μ and $\partial x^\mu/\partial\lambda$, specifying a point along the geodesic and its tangential velocity. The velocity vectors are constrained by an invariant scalar μ through

$$g_{\sigma\nu}\frac{\mathrm{d}x^\sigma}{\mathrm{d}\lambda}\frac{\mathrm{d}x^\nu}{\mathrm{d}\lambda} = \mu^2, \tag{4}$$

such that for $\mu^2 = 0$ one obtains the solutions for null-geodesics, $\mu^2 > 0$ space-like, and $\mu^2 < 0$ time-like **check the sign**. Specifying the three-vector $\partial x^i/\partial\lambda$ determines $\partial x^t/\partial\lambda$ for a chosen μ up to an ambiguity in the direction of time, which together with $g_{\mu\nu}$ and x^μ is sufficient to integrate the second-order ODE system. By default, we use the adaptive Tsitouras Runge-Kutta 5/4 algorithm.

how we find the initial conditions for velocity. Integration schemes, DifferentialEquations.jl and Tsit5 Redshift

★ E-mail: fergus.baker@bristol.ac.uk (FB)
[1] Open-source and available under MIT license at `https://github.com/astro-group-bristol/Gradus.jl`.

2.2 Solving for special orbits

How we find circular orbits, how we find ISCO, photon radius, event horizon

2.3 Tracing and constructing images

Also mention for the coronal sources

2.4 Transfer functions

Both 1d (Cunningham) and 2d (lag-energy) transfer functions, the methods used to solve them, and the quadrature integration schemes.

2.5 LNRF with Gram-Schmidt

How we derive the LNRF basis using Gram-Schmidt orthogonalization procedure, how we can use this to model corona

2.6 Disc emissivity

We can calculate emissivity / flux maps for discs using either Voronoi tesselation or some symmetric prescriptions

2.7 Covariant radiative transfer

From (2), with $a^\mu = f^\mu / m$.

WIP? will probably finish implementing this before writing the full paper

3 DESCRIPTION OF THE CODE

For Julia, AD is implemented in the ForwardDiff.jl package (Revels et al. 2016)

Multi-threaded, multi-CPU, optionally GPU decelerated. Speedup depends on choice of solver (fixed time step faster on GPU)

List of currently implemented metrics

Adding new spacetimes

Accretion disc geometries

Point functions for composable results

Performance vs e.g. Bambi's NK and other work

3.1 Simulation products

What we can export, and how they can be exported / used in e.g. XSPEC

4 TEST PROBLEMS

4.1 Integration stability

Energy conservation, deflection problem, shadow, tests for naked-singularities

4.2 Analytic solutions

All the special radii, the circular orbit energy

5 RESULTS

5.1 Iron line profiles

Bambi's various metrics and relline, self consistency between methods

5.2 Lag-frequency spectra

Ingram's code? Jiachen's code

5.3 Emissivity curves

Wilkins and Fabian with lamp post and moving corona

6 CONCLUSIONS

Outlook

We encourage the community to contact us with interesting problems that may be tackled using Gradus.jlas we are happy to assist with new applications of the code.

ACKNOWLEDGEMENTS

This work is supported by the UKRI AIMLAC CDT funded by grant EP/S023992/1.

We thank Cosimo Bambi and Jiachen Jiang for sharing their software for testing purposes. FB thanks Rosie Baker for her debugging assistance, and Gabriella Kirkpatrick for her insights and conversations.

Budgetary codes and sponsors including AIMLAC, UKRI

DATA AVAILABILITY

REFERENCES

Cunningham C. T., 1975, ApJ, 202, 788
Revels J., Lubin M., Papamarkou T., 2016, arXiv:1607.07892 [cs.MS]